Simplifying the Formal Verification of Safety Requirements in Zone Controllers through Problem Frames and Constraints based Projection

Simplifying the Formal Verification of Safety Requirements in Zone Controllers through Problem Frames and Constraints based Projection

Zhengheng Yuan, Xiaohong Chen*, Jing Liu, Yijun Yu*, Haiying Sun, Tingliang Zhou, Zhi Jin

***Abstract*—Formal methods have been applied widely to verifying the safety requirements of Communication-Based Train Control (CBTC) systems, while the problem situations could be much simplified. In industrial practices of CBTC systems, however, huge complexity arises, which renders those methods nearly impossible to apply. In this paper, we aim to reduce the state space of formal verification problems in Zone Controller, a sub-system of a typical CBTC. We achieve the simplification goal by reducing the total number of device variables. To do this, two *projection* methods are proposed based on Problem Frames and constraints, respectively. The Problem Frames based method decomposes the system according to sub-properties through functional decomposition, whilst the constraints based projection method removes redundant variables. Our industrial case study demonstrates the feasibility though an evaluation, confirming that these two methods are effective in reducing the state spaces of complex verification problems in this application domain.**

***Index Terms*—Problem Frames Approach; Projection; Zone Controller; Constraints; Formal Verification**

I. Introduction

SAFETY is the key requirement for train control systems to avoid collisions. Unlike traditional train control systems that use fixed blocks, the new generation of Communication-Based Train Control (CBTC) systems use moving blocks instead, which allows more trains to run on the same track simultaneously. Since formal methods have successful application in rail transit systems, e.g., Paris Metro Line 14 back in 1998 [1], their uses have been recommended by various safety domain standards including EN50128 [2] and EN50129 [3].

Formal verification has also been applied, with limited success, to a sub-system of CBTC, namely Computer Interlocking [4]. However, such a verification was feasible only because a table of boolean equations could be used to greatly simplify the interlocking sub-system. Apart from this limited success in verifying the interlocking sub-system, other parts of the CBTC had not been verified successfully, even for a resourceful company, CASCO Signal Ltd.[1]. In 2012, CASCO has been qualified for SIL-4, the highest level of safety standards, after three years of applying formal modeling throughout the development of Zone Controllers (ZC). By the end of such formal development, however, they failed to verify ZC using the Prover model checker [5] associated with SCADE [6] or any other existing formal verification tools.

Through an analysis of such failures, it was our belief that the cause lies fundamentally in the complexity of ZC. ZC is designed to help trains move into the right position. Its main task is to tell how far the track in front of a train is safe for its running, or professionally called, "claiming the Movement Authority (MA)". According to [7], ZC can be divided into about 20 sub-systems, and there will be more modules when being implemented. These modules interact with different devices. By devices, we mean the entities that will interact with ZC (sub-)systems. They not only refer to the equipment such as signal lights (called signals in the domain), but also including existing software such as Computer Interlocking system, and virtual nodes such as Virtual Train Protection (VTP), which is a logical enlargement of train. According to IEEE standard 1474.1 [8], a ZC can interact with more than a thousand such devices. Since devices are represented by sets of variables in the formal verification system, one device could have thousands of possible values. All these values make the model of a system too huge to be computed by existing verifiers, forming a classical state explosion problem which has been the blocker for many applications of formal methods.

Although state explosion problems could not be solved directly, one could carry out the verification on smaller models if the original problem could be partitioned into much smaller sub-problems. As long as the verification results of sub-problems do not affect each other, their results could be combined to decide the outcome of the verification of the entire system.

Further analysis revealed to us that the complexity, or the size of composed state space, depends largely on the number of variables representing the actually used devices. The interactions between these devices and the systems manifest as communications, whilst the constraints about the variables are obtained from the domain knowledge. For example, there are two kind of devices, block and branch. Block is a segment of a track. It is base unit of the track in ZC. Branch is an access that composed by all connected blocks at current state [9]. The blocks are indexed using a block coordinate, whilst the branches are indexed using a branch coordinate. Any position on the track can be located using both coordinates. There are constraints among the coordinate variables.

Zhengheng Yuan, Xiaohong Chen, Jing Liu and Haiying Sun are with Shanghai Key Laboratory of Trustworthy Computing, East China Normal University.

Yijun Yu is with School of Computing and Communications, The Open University.

Tingliang Zhou is with CASCO Signal Ltd..

Zhi Jin is with Key Laboratory of High Confidence Software Technologies, Ministry of Education, Institute of Software, School of EE & CS, Peking University.

Xiaohong Chen and Yijun Yu are the corresponding authors.

[1]The company has above 70% market share of signaling systems in China.

According to these observations, we propose two projection methods for the variables. The Problem Frames (PF)-based projection decomposes the problem into smaller sub-problems dealing with collaborating relations, whilst the constraints-based projection reduces the number of variables in addressing some of the constraints, even when all the variables are based on valid sampling data.

In this paper, we hypothesize that properly reducing the number of variables could reduce the size of state space without changing the results of verification. By defining the two projection methods, we have shown the feasibility using real life cases. Experiments have also been carried out to demonstrate and confirm our hypothesis. Based on such experiences, we suggest how to use our methods effectively.

The remainder of the paper is organized as follows. Section II defines ZC and its verification problem, and proves the basic hypotheses in our work. Section III and section IV present the PF-based projection and constraints-based projection methods, respectively. Section V presents an industrial case study using these two projection methods, with the results evaluated in section VI. Section VII compares to related work and section VIII concludes.

II. Problem Description and Hypothesis

In this section, firstly we introduce the Zone Controller (ZC) and its verification problem, and use problem diagrams in the PF approach [10] to represent them. Then we present the basic hypotheses of our methods.

A. Problem of Zone Controller

A ZC system is a component of CBTC responsible for controlling a part of the track among several stations. Its main purpose is calculating Movement Authority (MA) for all trains under its control. The MA is calculated by the information that ZC received from the Vehicle On-Board Computer (VOBC), the Automatic Train Supervision (ATS) and the Computer Interlocking system (CI), as well as many sensors along the track. In our collaborative project with CASCO Ltd., a ZC system also has other functions such as calculating the safe location of the train, sorting the trains in the area, and updating the track occupancy status.

According to the above descriptions, we define all the signaling devices that can interact with the ZC as parts of its environment, depicted by a problem diagram [10] in Figure 1. A ZC system is a machine M which interacts with its environment E, satisfying the safety requirements R. Recording the interactions as IS, we formally define the problem as a four-tuple: $P =< M, E, IS, R >$ [11].

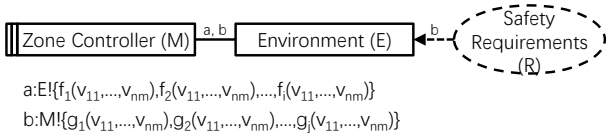


Fig. 1. Problem Diagram of Zone Controller

In the problem description, M is the software to be developed. E consists of many devices (including other systems in CBTC and sensors along the track). Each state of device can be indicated by many more variables. For example, on the track, there is a type of devices named `point` which needs 5 variables to describe: one `state` variable indicates that its state is in one of the values of `normal`, `reverse`, or `unknown`; the other four variables, namely `nextidx_1`, `nextidx_2`, `nextidx_3`, `nextidx_4`, represent the IDs of the blocks next to it.

Therefore, the formal description of E is defined by these devices in the environment of the ZC, with each device defined by a set of variables:

***Definition** 1:* Environment E is a set of devices, where each device interacts with ZC, represented by a set of variables v_{ij}, i.e., $E = \{D_i \mid 1 \leq i \leq n\}$, $D_i = \{v_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq m\}$.

Since there are too many devices to be presented in limited space, instead of enumerating them one by one, Definition 1 uses D_i, so does Figure 1.

Depending on the initiator, all the interactions between M and E can be classified into two sets, a and b (following PF's convention). Interactions in a are the phenomena initiated by the environment E ('!' means 'control'), whilst interactions in b are initiated by the machine M. The interactions in a and b are functions of the variables $v_{ij}(1 \leq i \leq n, 1 \leq j \leq m)$ in E, denoted by $f_i(V_1, V_2, \cdots, V_n)$ and $g_j(V_1, V_2, \cdots, V_n)$.

R is a set of safety requirements. In this paper, we define a safety requirement as a condition that should be met by the machine to make sure that people is safe. An example of safety requirement is "a train should never crash on another". Generally speaking, R is often written in natural language, and can be decomposed into sub-requirements (sub-properties). For instance, the above example can be decomposed into sub-properties: (1) ZC has to ensure that there does not exist another train in MA, and (2) VOBC has to ensure that the speed of the train never exceeds speed curve. In most cases, the sub-properties can be found in domain standards such as IEEE 1474.3 [12]. We record the relation between safety requirements (R) and their sub-properties ($P_i, 1 \leq i \leq n$) as $R = \{P_1, P_2, \cdots, P_i, \cdots, P_n\}$.

B. Verification Problem of Zone Controller

To verify ZC as shown by the problem diagram in Figure 2, we construct a system of verification machine VM, which monitors the inputs and outputs of the machine M, and checks whether the outputs satisfy safety requirements in all reachable states (i.e. verification requirements R'), and finally generates a verification report (VR). It is noticed that VR is also a problem domain, which is the result of problem that excluded it as a problem domain. However, in the PF approach, the to-be-built data storage can be modelled as a type of problem domains called "Designed Domains" denoted by a rectangle with a single vertical line in the problem diagram. So the environment of ZC verification problem E' is composed by the environment of the ZC (E), the ZC itself (M) and the verification report (VR): $E' = E \cup \{M\} \cup \{VR\}$.

The interactions between the environment are as follows. According to the ZC problem, the interactions between M and

E are in a and b while a is the inputs and b is the outputs from the perspective of M. The VM should use a and b accordingly to judge whether R' is satisfied. Therefore, the interactions between E and VM are in a' which actually monitors the interactions in a, whilst the interactions between VM and M are in b' which reflect b. We can see that a' and b' are actually a and b except that they have different initiators or receivers. Therefore, there is no need to include a and b in the problem diagram. For the interactions between VM and VR, we designate a boolean variable $bValid$ and a string variable $cExample$ in c to represent the verification results and the counter-example, respectively.

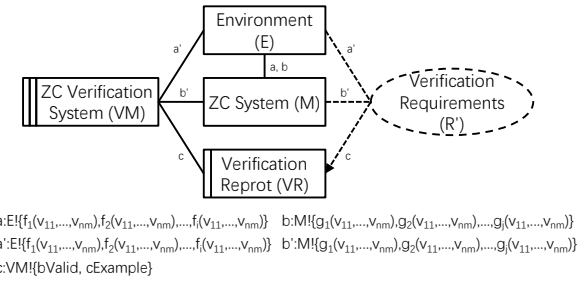


Fig. 2. Problem Diagram of ZC Verification System

To sum up, the verification system of ZC includes the following parts: the verification system VM, its environment E', interactions in a', b', and c and requirements R':

***Definition** 2:* A verification problem is defined as a four-tuple $VP =< VM, E', IS', R' >$, where VM is the verification system to verify M. E' is the environment of VM, which is the union of E, M and VR. VR is the verification report of the verification problem, which consists of a boolean variable $bValid$ and a string type counter example $cExample$ if $bValid == false$. $IS' = a' \cup b' \cup c$ is the set of interactions between VM and E', where c is a set including $bValid$ and $cExample$. R' is the set of verification requirements, which are to verify whether M satisfies R.

The PF descriptions of ZC and its verification problems are at a high level. As to their low level implementations, an implementation language is required. In the railway transportation area, especially in our collaboration project with CASCO Ltd., SCADE [13] is widely used. In this paper, we choose Scade[2] as implementation language, designating its semantics to be our ZC problem and verification problem's.

C. Hypotheses and Theoretical Validation

The two projection methods proposed in this paper is built on a hypothesis. That is, reducing the number of variables in VP could decrease the time cost of the verification without altering the results. This hypothesis can be divided into two sub-hypotheses, as listed below.

***Hypothesis** 1:* Reducing the number of variables can decrease the time cost of verification.

According to [14], a system's state space could be computed by the product of the number of possible values of variables therein. Our verification problem VP includes the values E' (E, M, VR), VM, IS' and R'. We claim that the state space of VP can be determined by the variables of E and M. The reasons are as follows. Firstly, as VR is a designed domain, its variables do not affect the size of VP, therefore it should be excluded in estimating the time complexity. Secondly, the interactions between VM and E, VM and M are actually interactions between M and E. These interactions are used to monitor and control variables in the environment, which means that these interactions are about the variables in E. Finally, the variables in safety requirements R' or sub-properties to be monitored are actually expected effects on the environment. In fact, they are the variables of declared in the environment or with the need of computation.

Since M is a black box, one could not tell the private variables hidden inside. Therefore, one could only reduce variables in the environment. In this case, it is obvious that reducing the number of variables could reduce the state spaces, leading to less time cost.

***Hypothesis** 2:* Reducing dependent variables does not change the result of verification.

By dependent variables we mean the variable could be expressed by other variables since the constraints in ZC are mainly in terms of equations, For example, suppose $iHead$ and $iTail$ represent the location of the head and tail of a train, and $iLength$ is its length, they have a constraint (suppose $iHead$ is larger than $iTail$), $iHead - iTail = iLength$. In this case $iLength$ can be expressed by $iHead$ and $iTail$, as well as the other variables. One can say that $iHead$, $iTail$ and $iLength$ are dependent variables.

To validate this hypothesis, it is to verify that the semantic of VP does not change before and after the reduction. According to [13], two expressions in Scade can be considered as equivalent in semantics if their types, clocks, values are equal. Here type means the variable type such as $integer$. Clock is a logic concept which presents a sequence of time. Clock equivalence refers to "at the same time". Value in Scade is a function of time. When a variable is replaced with an expression, it is ensured that at any time, the variable and expression share the same type and value. In this sense, the semantics of VP before and after the replacement are the same. Therefore, the reduction will not change the verification result.

III. Problem Frames Approach based Projection

The PF approach is suitable for decomposing a problem into smaller sub-problems through generalized projection methods [15]. A projection requires an object and a dimension. Here, the projected object is the verification problem VP, and the projected result is a set of smaller verification problems $VP_1, VP_2, \cdots, VP_n$. The difficulty in applying this method directly to the ZC verification problem lies in how to find a suitable projection dimension. The projection dimension should be an element that is related to every element defined in VP including VM, E' (E, M, VR), R' and IS'.

In the ZC problems, R is divided into sub-requirements as sub-properties $\{P_1, P_2, \cdots, P_n\}$. For example, the top level requirement of ZC is "the trains should run safely", which

[2]According to Esterel Ltd., SCADE refers to the development environment whilst Scade refers to the modelling language.

can be divided into two requirements: "the train should not crash on anything", and "the train should not derail" [16].

For each sub-property P_i, one could obtain different types of M (M_i) and related environment E_i from IEEE standard 1474.1 [8] and 1474.3 [12]. They can be listed in Table I in which builds a mapping among E_i, M_i and P_i. The relationship among them is that M_i runs in E_i and should satisfies P_i.

The process to obtain Table I is as follows. Firstly, one has to find the related chapters for R in Standard 1474.1. In many cases, the standard provides several situations that the function has to deal with. For example, a requirement "safe train separation", in chapter 6.1.2, lists 8 situations on page 18 with "The movement authority limit shall be the most restrictive of the following:". These situations lead to sub-properties P_i ($1 \leq i \leq 8$). Secondly, find functions that are related/close to the requirement R in Standard 1474.3. For example, the function related to the requirement "safe train separation" is in chapter 6.3-limit of movement protection and target point determination. By mapping the functions in chapter 6.3 with P_i, we obtain M_i. Finally, according to scenarios and design details in each company, the domain experts could get E_i.

One can see that the process is not easy to be standardized because it is domain knowledge dependent. But for the domain experts, it is not difficult to get the mapping. Different companies may have different results. In order to keep focus, in this paper, we only use Table I as part of the inputs.

TABLE I
RELATIONS AMONG SUB-SYSTEMS, SUB-PROPERTIES AND SUB-ENVIRONMENTS

Sub-system	M_0	M_1	$\cdots$	M_i
Sub-property	P_0	P_1	$\cdots$	P_i
Sub-environment	E_0	E_1	$\cdots$	E_i

Additionally, P_i can correlate to IS', VM and VR. Since E_i is determined, the shared phenomena IS'_i are also decided. The IS'_i defines a new machine VM. Finally VR can get its value VR_i after all other elements have been built.

With these relations in-between, P_i has relation to all elements in VP so that it can be used as a projection dimension. Therefore, projection is performed along with the sub-property P_i. Adopting an expression similar to relational algebra, this formula of problem projection can be defined as follows.

***Definition** 3:* A verification problem VP can be projected according to a sub-property P_i, after which VP becomes VP_i:

$$\begin{aligned} VP_i &= \pi_{P_i}(VP) \\ &=< \pi_{P_i}(VM), \pi_{P_i}(E'), \pi_{P_i}(IS'), \pi_{P_i}(R') > \end{aligned}$$

where π_{P_i} is the auxiliary projection operator which could be defined for different elements in VP as follows.

- Verification system projection operator for VM

  $$\pi_{P_i}(VM) = VM_i$$

  The projection defines a new sub-verification system recording VM_i.
- Environment projection operator for E'

  $$\begin{aligned} \pi_{P_i}(E') &= E'_i \\ &= \pi_{P_i}(E) \cup \{\pi_{P_i}(M)\} \cup \{\pi_{P_i}(VR)\} \\ &= E_i \cup \{M_i\} \cup \{VR_i\} \end{aligned}$$

  The projection operator chooses M_i and E_i related to P_i according to table I, and defines VR_i as the result of this property verification.
- Interaction projection operator for IS'

  $$\begin{aligned} \pi_{P_i}(IS') = \quad & \{X | X \in IS, \\ & receiver(X) \cup initiator(X) \in E'\} \end{aligned}$$

  where receiver(X) returns the receiver of X, initiator(X) returns the initiator of X. This operator keeps interactions initialized or received by domains in E_i.
- Requirement projection operator for R'

  $$\pi_{P_i}(R') = P'_i$$

  P'_i is to verify whether M_i satisfies P_i.

To sum up, the verification problem obtained after the projections VP_i describes four types of elements covering the system to be verified: the environment, the verification system, the verification requirements, and the interactions. Amongst them, the environment obtained after projection is the subset of original environmental properties, the safety requirements are the subset of original problem safety requirements. The number of input variables of the verification system also reduces with the environment through the interactions. Thus after the projection, the complexity of such verification system will decrease, whilst the state space generated by the verification formula will shrink in size.

IV. CONSTRAINTS BASED PROJECTION

Constraints-based projection aims to analyze the constraints among variables and tries to find out which redundant variables can be removed from the system model. This kind of projection is different from PF-based method because it is at the variable level while the former is at the functional level. This section classifies the constraints in ZC, selects the projection dimension, and defines the projection.

A. Constraints Classification

According to the number of devices variables belong to, constraints can be classified into internal and external relations. Internal relations are about variables from the same device, and external ones are about variables from different devices. Most of the constraints in ZC systems take the form of equations, denoted as: $f(x_1, x_2, \cdots, x_p) = g(y_1, y_2, \cdots, y_q)$, where $\{x_i\} \cap \{y_j\} = \emptyset$, $1 \leq i \leq p$, $1 \leq j \leq q$, and f, g are functions of variables on left side or right side.

In this paper, we only deal with the constraints in the form of equations. From the perspective of algebraic equations, it seems that the left and right hand sides are in the same position, which means the variables in both sides could substitute each other. For example, with the same constraint $iHead - iTail = iLength$ in Section II, $iLength$ can be replaced by $iHead - iTail$, and $iHead$ can be substituted by $iTail + iLength$.

In reality, however, the positions of variables on the two sides may not be the same. Due to the efficiency of equation solvers and cost problem, variables on the left-hand (resp. right-hand) side can be used to replace the variables on the right-hand (resp. left-hand) side, but the right (resp. left) ones

cannot be used to replace the left (resp. right) ones. For example, there are two coordinates systems in ZC presenting each location on the track. One is indexed by $blocks$, and the other one is indexed by $branches$. Theoretically, the $block$ coordinates could transform to the $branch$ coordinates and vice versa. But in fact, the transformation from branch to block costs too much. Once needed, one always transforms the $block$ coordinates to the $branch$ coordinates, but not the other way around. According to the constraints between these two coordinate systems, it would cost huge amount of computation to transform from $branch$ to $block$, which leads to more serious state explosion problems. Therefore, for this kind of constraints only one side variables can be substituted by the other side ones.

In order to distinguish these two kinds of equality constraints, we use two different notations. The first one still uses "$=$", while the second uses "$\Rightarrow$" instead.

B. Projection Dimension Selection

Given a set of constraints, we need to choose a set of variables as the project dimension, which means to compute how many variables are needed and how to choose them from all the variables.

For the first question, according to algebraic equation solution, each independent equation about the set of variables can be used to remove one variable from the set. When one has n variables and m equations, if $n \leq m$, all variables can get its value unless there exists a conflict. On the other side, if $n > m$, only $n - m + 1$ variables are needed. So the projection dimension is about $n-m+1$ free variables, denoting as $V = \{v_1, v_2, \cdots, v_{n-m+1}\}$. The conflict mentioned above means that the equations have no solution. This means that there must be something wrong in the constraints. Since the constraints are from the real world, the ideal constraints can provide an input space which is exactly the same as real world. In these cases, all constraints should have at least one solution in theory. The only challenge would be writing down such constraints. When facing this challenge, one has to discuss with domain experts about the constraints in order to find where mistakes were made.

For the second question, one can order these variables by their importance. The more importance they have, the higher priority. Suppose that one has a list of decreasing priority, the first $n - m + 1$ variables could be selected as projection dimension. Challenge is now how to build such a priority list. In practices, we summarize the following guidelines:

- Priorities could be given by domain experts.
- The variables that never occur in any constraint have the highest priority.
- For the "$\Rightarrow$" constraints, any variable on the right hand side can only be replaced by variables on the left exist. Therefore, one could only choose the variables on the left-hand side.
- For "$=$" constraints, one can count the occurrences of each variable. The more occurrences, the higher priority it has. Another strategy is to compare the value range of variables and choose the one with possibly fewer values.

To reduce the subjectiveness of the selection process, we develop Algorithm 1 to automatically obtain the priority list. The algorithm takes a variable set and a constraint set as inputs, and outputs a priority list. The basic idea is as follows. First, it finds the variables that do not exist in any constraints. These variables are considered as highest priorities. Secondly, it considers all the "$\Rightarrow$" constraints to obtain all the left hand side variables. Then it considers the "$=$" constraints to choose the more frequently occurred variables. If the occurrences of two variables are the same, choose the one with fewer values. Finally, if they have the same value range, select one randomly. The detailed process is shown in Algorithm 1. The complexity of the algorithm is $O(n^2)$.

Algorithm 1 Priority List Generation Algorithm

Input: variables VS, constraints CS;
Output: priority list PL;
1: Begin:
2: $PL \Leftarrow$ all variables in VS but not mentioned in CS;
3: **for** each $c \in CS$ **do**
4: **if** c is an equation with $\Rightarrow$ **then**
5: count the occurrence numbers N of variables on left side;
6: **else**
7: count the occurrence numbers N of all variables;
8: **end if**
9: **end for**
10: sort the variables by their occurrence numbers N;
11: **if** the first $n - m + 1$ variables can be selected **then**
12: $PL \Leftarrow$ the first $n-m+1$ variables; // then the priority list is consisted of these variables
13: **else**
14: **for** each variable that share the same occurrence number **do**
15: calculate the value space of the variable; // choose $n - m + 1$ variables to consist of PL
16: **end for**
17: order the variables by the value space;
18: $PL \Leftarrow$ the first $n - m + 1$ variables;
19: **end if**
20: return PL;
21: End;

C. Constraints based Projection Definition

Having obtained the projection dimension $V = \{v_1, v_2, \cdots, v_{n-m+1}\}$, the constraints based projection can be defined. Back to the verification problem, among the four elements, E' and R' will not change after projection, only VM and IS' will be projected into smaller ones by V. The exact definition is shown as follows.

Definition 4: The verification problem VP can be projected according to a variable set V, and after the projection VP becomes VP_V, which can be expressed as follows:

$$\begin{aligned} VP_V &= \pi_V(VP) \\ &=< \pi_V(VM), E', \pi_V(IS'), R' >, \end{aligned}$$

where two auxiliary projection operators for different elements in VP needs defining:

- Verification system projection operator for VM,
$$\pi_V(VM) = VM_V,$$
here VM_V checks whether M satisfies R, and can be named by the users.
- Interaction projection operator for IS',
$$\pi_V(IS') = IS'_V,$$

where $IS'_V = \{y | y = repPhe(x, phyz), x \in IS'\}$, $repPhe(x, phyz)$ returns an interaction obtained by replacing the variable in interaction x with $phyz$, which $phyz$ can be computed by Algorithm 2.

In the interaction project operator, the variable replacement could be done automatically. We design Algorithm 2 to compute every phenomenon in the interaction set IS' with variables in the projection dimension V. It tries to solve all constraints and present all phenomena after projection. By solving, we mean that it could be expressed by variables in projection dimension or existing solved variables. Generally speaking, the algorithm traverses all unsolved constraints continuously, and if a constraint can be solved, it will be replaced with an expression in all constraints. The complexity of this algorithm is $O(n)$.

Algorithm 2 Computation of Phenomena after Projection

Input: constrains C, projection dimension V;
Output: phenomena after projection R;
1: Begin:
2: **while** C is not empty **do**
3: choose c from C randomly;
4: **if** c is an equation with $\Rightarrow$ **then**
5: **if** all variables on the left of c are in D **then**
6: **for** each v on the right of c **do**
7: add expression of v by D to R; //v is solved
8: $D = D \cup \{v\}$; //v can be used to solve other variables
9: **end for**
10: $c = C - c$;
11: **end if**
12: **else**
13: **if** there is only one variable v does not belongs to D **then**
14: add expression of v by D to R;
15: $D = D \cup \{v\}$; //v is solved
16: $C = C - \{c\}$; //v can be used to solve other variables
17: **end if**
18: **end if**
19: **end while**
20: End;

V. Case Study

In this section, we present a sub-system of ZC from a collaborated project, called CAL_EOA. It is used to calculate MA for each train, i.e., a permission for a train to move to a specific location with supervision of speed. We apply our two methods to demonstrate their feasibility. The project track plan is "block number is 6, point number is 2, and the max number of other devices is 4". This scale track plan is a base verification unit in their plan that trying to use formal methods in development process. The problem description is as follows.

Calculating the MA means to calculate the start point and end point. The start point is the minimum train head, and the end point is called end of authority (EOA), which means the maximum range of authorized train movement. The calculation of EOA relies on the trains (TR), track consisting of blocks (BL) and branches (BR), signals (SI), buffer zone (BZ), overlap (OL), traffic direction (TD), and ZC boundary (ZB). It has to be mentioned that one of the train information, location not only refers to the coordinates, but also a VTP.

The above devices have 31 variables (for brevity, all variables here are simplified). They are listed in table II. Among these variables, 11 constraints are listed in Table III, where $c_1 - c_8$ are internal constraints of train, c_9 and c_{10} are internal constraints of block and branch. Constraint c_{11} is an external relation between BL and BR.

TABLE II
Variables in CAL_EOA

Device	Variable	Explanation
TR	THaPo	The Position of max Train Head
	THiPo	The Position of minimum Train Head
	TTaPo	The Position of max Train Tail
	TTiPo	The Position of minimum Train Tail
	VHaPo	The Position of max VTP Head
	VHiPo	The Position of minimum VTP Head
	VTaPo	The Position of max VTP Tail
	VTiPo	The Position of minimum VTP Tail
	TRLen	Train Length
CAL_EOA	EOATyp	EOA Type
	EOAPos	EOA Position
BL	BLpID	Block ID on Up Direction
	BLnID	Block ID on Down Direction
	BLLen	Block Length
	BLiBR	The Branch ID of that the Block locates
BR	BRpID	Branch ID on Up Direction
	BRnID	Branch ID on Down Direction
	BRLen	Branch Length
SI	SIPos	Signal Position
	SIDir	The Direction of the Signal
	SISta	The State of the Signal
ZCB	ZBPos	The Position of the ZC Boundary
	ZBDir	A Direction of the ZC Boundary
TD	TDPos	The Position of the Traffic Direction
	TDDir	The Direction of the Traffic Direction
	TDSta	The State of the Traffic Direction
BZ	BZPos	The Position of the Buffer Zone
	BZDir	The Direction of the Buffer Zone
	BZSta	The State of the Buffer Zone
OL	OLPos	The Position of the Overlap
	OLDir	The Direction of the Overlap

According to ISO Standard 1474.3 [12], the safety requirement of EOA calculation is divided into 8 sub-properties. Each sub-property involves different environment as listed in Table IV.

A. Problem Diagrams of CAL_EOA & its Verification Problem

From the problem description of CAL_EOA, a problem diagram of CAL_EOA is obtained in Figure 3. To clearly show the interactions in the problem diagram, we list the interactions in Table V. From them, the problem description of EOA is:

$$P_{CAL\_EOA} =< CAL\_EOA, E, IS, R >$$

where

- CAL_EOA is the system to be built;
- E includes trains (TR), block (BL), branch (BR), signal (SI), ZCB (ZB), traffic direction (TD), buffer zone (BZ) and overlap (OL), i.e.,
$$E =< TR, BL, BR, SI, ZB, TD, BZ, OL >$$
- IS is the interaction set between CAL_EOA and E.
$$IS = \{ita_1, ita_2, \cdots, ita_{31}\}$$
where ita_i $(1 \leq i \leq 31)$ is defined in Table V.
- R is the safety requirement, with a nature language description: "the range between the minimum head of the train and the EOA point its output cannot have any

TABLE III
DESCRIPTIONS OF CONSTRAINTS

Constraint	Description	Equation
c_1	The distance between $THiPo$ and $TTiPo$ is $TRLen$	$THiPo - TTiPo = TRLen$
c_2	The distance between $THaPo$ and $TTaPo$ is $TRLen$	$THaPo - TTaPo = TRLen$
c_3	The distance between $THaPo$ and $THiPo$ is a constant C_1	$THaPo - THiPo = C_1$
c_4	The distance between $TTaPo$ and $TTiPo$ is a constant C_2	$TTaPo - TTiPo = C_2$
c_5	$THiPo$ and $VHiPo$ are the same point	$THiPo = VHiPo$
c_6	The distance between $VHaPo$ and $VHiPo$ is a constant C_3	$VHaPo - VHiPo = C_3$
c_7	The distance between $VTaPo$ and $TTaPo$ is a constant C_4	$VTaPo - TTaPo = C_4$
c_8	The distance between $TTiPo$ and $VTiPo$ is a constant C_5	$TTiPo - VTiPo = C_5$
c_9	The block ID in up direction $BLpID$ and down direction $BLnID$ are same	$BLpID = BLnID$
c_{10}	The branch ID in up direction $BRpID$ and down direction $BRnID$ are same	$BRpID = BRnID$
c_{11}	The length of a branch can be calculated by add all length of the blocks belong to it	$f(BLLen, BLiBR, BRnID) \Rightarrow BRLen$

TABLE IV
DESCRIPTIONS OF DIFFERENT SUB-PROPERTIES

Sub-property	Description	Involving environment	Sub-system
P_1	There cannot exist any end of track between the minimum train head and EOA.	$E1 = \{TR, BL, BR\}$	M_1
P_2	There cannot exist any other CBTC-equipped train between the minimum train head and EOA.	$E2 = \{TR, BL, BR\}$	M_2
P_3	There cannot exist any uncontrolled point between the minimum train head and EOA.	$E3 = \{TR, BL, BR\}$	M_3
P_4	There cannot exist any ZC boundary between the minimum train head and EOA.	$E4 = \{TR, BL, BR, ZB\}$	M_4
P_5	There cannot exist any discontinuous traffic direction between the minimum train head and EOA.	$E5 = \{TR, BL, BR, TD\}$	M_5
P_6	There cannot exist any buffer zone which does not allow the train to pass between the minimum train head and EOA.	$E6 = \{TR, BL, BR, BZ\}$	M_6
P_7	There cannot exist any branch break between the minimum train head and EOA.	$E7 = \{TR, BL, BR, ZB\}$	M_7
P_8	There cannot exist any overlap which does not allow the train to pass between the minimum train head and EOA.	$E8 = \{TR, BL, BR, OL\}$	M_8

TABLE V
INTERACTIONS OF CAL_EOA AND ITS VERIFICATION SYSTEM

Interaction ID	Initiator	Receiver	Phenomenon
ita_1 (ita_{32})	TR	CAL_EOA (VM)	THaPo
ita_2 (ita_{33})	TR	CAL_EOA (VM)	THiPo
ita_3 (ita_{34})	TR	CAL_EOA (VM)	TTaPo
ita_4 (ita_{35})	TR	CAL_EOA (VM)	TTiPo
ita_5 (ita_{36})	TR	CAL_EOA (VM)	VHaPo
ita_6 (ita_{37})	TR	CAL_EOA (VM)	VHiPo
ita_7 (ita_{38})	TR	CAL_EOA (VM)	VTaPo
ita_8 (ita_{39})	TR	CAL_EOA (VM)	VTiPo
ita_9 (ita_{40})	TR	CAL_EOA (VM)	TRLen
ita_{10} (ita_{41})	CAL_EOA	TR (VM)	EOATyp
ita_{11} (ita_{42})	CAL_EOA	TR (VM)	EOAPos
ita_{12} (ita_{43})	BL	CAL_EOA (VM)	BLpID
ita_{13} (ita_{44})	BL	CAL_EOA (VM)	BLnID
ita_{14} (ita_{45})	BL	CAL_EOA (VM)	BLLen
ita_{14} (ita_{46})	BL	CAL_EOA (VM)	BLiBR
ita_{16} (ita_{47})	BR	CAL_EOA (VM)	BRpID
ita_{17} (ita_{48})	BR	CAL_EOA (VM)	BRnID
ita_{18} (ita_{49})	BR	CAL_EOA (VM)	BRLen
ita_{19} (ita_{50})	SI	CAL_EOA (VM)	SIPos
ita_{20} (ita_{51})	SI	CAL_EOA (VM)	SIDir
ita_{21} (ita_{52})	SI	CAL_EOA (VM)	SISta
ita_{22} (ita_{53})	ZB	CAL_EOA (VM)	ZBPos
ita_{23} (ita_{54})	ZB	CAL_EOA (VM)	ZBDir
ita_{24} (ita_{55})	TD	CAL_EOA (VM)	TDPos
ita_{25} (ita_{56})	TD	CAL_EOA (VM)	TDDir
ita_{26} (ita_{57})	TD	CAL_EOA (VM)	TDSta
ita_{27} (ita_{58})	BZ	CAL_EOA (VM)	BZPos
ita_{28} (ita_{59})	BZ	CAL_EOA (VM)	BZDir
ita_{29} (ita_{60})	BZ	CAL_EOA (VM)	BZSta
ita_{30} (ita_{61})	OL	CAL_EOA (VM)	OLPos
ita_{31} (ita_{62})	OL	CAL_EOA (VM)	OLDir
ita_{63}	VM	VP	bValid

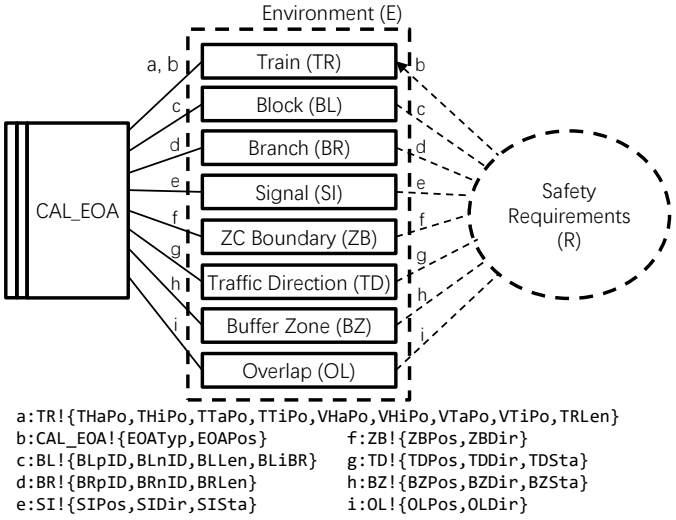


Fig. 3. Problem Diagram of CAL_EOA

point belonging to one of the eight kind of unsafe point defined in Table IV".

As defined in section II, we obtain the problem diagram of the CAL_EOA verification problem in Figure 4. Here we does not present the interaction between E and M because it is not necessary in the method. The verification problem of CAL_EOA (VP) is defined as:

$$VP =< VM, E', IS', R' >$$

where

- VM is the verification machine for checking whether CAL_EOA satisfies R'.
- $E' = \{CAL\_EOA\} \cup E \cup \{VR\}$.
- $IS' = \{ita_{32}, \cdots, ita_{63}\}$, where each interaction is defined in Table V.
- R' is "to verify that CAL_EOA satisfies the safety requirements R".

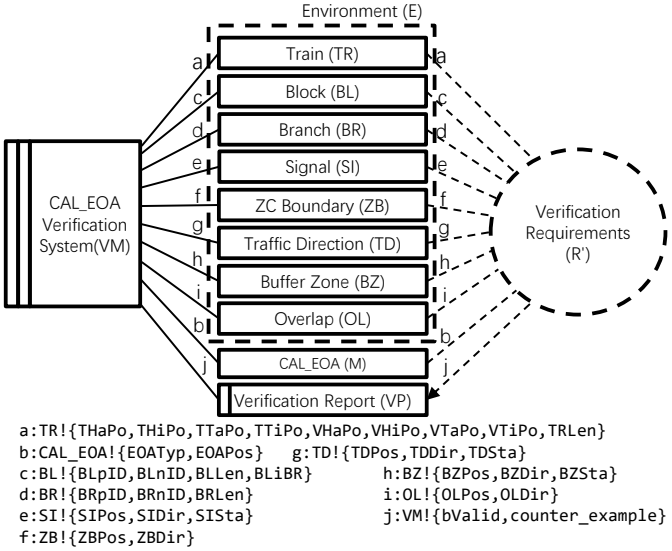


Fig. 4. Problem Diagram of CAL_EOA Verification System

B. Projection based on Problem Frames Approach

In the application of our PF based projection method, the projection object is VP, and projection dimension is the sub-property given in Table IV. There are 8 sub-properties, so we get 8 projection dimensions. For each dimension $P_i, (1 \leq i \leq 8)$, we could get a sub-problem VP_i by using projection. In the following, we will take P_8 as the projection dimension, and show how we get VP_8.

$$\begin{aligned} VP_8 &= \pi_{P_8}(VP) \\ &=< \pi_{P_8}(VM), \pi_{P_8}(E'), \pi_{P_8}(IS'), \pi_{P_8}(R') > \end{aligned}$$

According to projection operators definition, we obtain:

- $\pi_{P_8}(VM) = VM_8$
- $\pi_{P_8}(E') = E'_{ij} = E_8 \cup \{CAL\_EOA\} \cup \{VR\}$ $= \{TR, BL, BR, OL, CAL\_EOA, VR\}$
- $\pi_{P_8}(IS') = \{ita_{32}, \cdots, ita_{40}, ita_{43}, \cdots, ita_{49}, ita_{61}, ita_{62}, ita_{63}\}$
  where each interaction is shown in Table V.
- $\pi_{P_8}(R') = P_8$

Similarly, we could get the other 7 sub-problems $VP_1 - VP_7$. After the projection, we put these 8 sub-problems in SCADE, and use its built-in verifier to verify them. They successfully pass the verification within at most 147 seconds.

C. Projection based on Constraints

For constraints based application, we should first choose the projection dimension V. According to table II and III, there are 31 variables and 11 constraints, so a projection dimension should include 21 variables. Using Algorithm 1, a priority list is obtained. We choose the first 21 variables to form V:

$$\begin{aligned} V = \quad & \{EOATyp, EOAPos, SIPOs, SIDir, SISta, \\ & ZBPos, ZBDir, TDPos, TDDir, TDSta, BZPos, \\ & BZDir, BZSta, OLPos, OLDir, BLiBR, BLLen, \\ & BRnID, THiPo, TRLen, BLnID\} \end{aligned}$$

where the first 15 variables are independent, and the last 6 ones are chosen from dependent variables.

According to the definition of constraint-based projection, we project the VP as follows:

$$\begin{aligned} VP_V &= \pi_V(VP) \\ &=< \pi_V(VM), E', \pi_V(IS'), R' > \end{aligned}$$

- $\pi_V(VM) = VM_V$
- $\pi_V(IS') = IS_V = \{ita_{33}, ita_{40}, ita_{41}, ita_{42}, ita_{44}, ita_{45}, ita_{46}, ita_{48}, ita_{50}, \cdots, ita_{73}\}$
  where each interaction is shown either in Table V or VI.

TABLE VI
INTERACTIONS WITH PHENOMENA CHANGED AFTER PROJECTION

Interaction ID	Initiator	Receiver	Phenomenon
ita_{64}	TR	VM	THiPo+C_1
ita_{65}	TR	VM	THiPo-TRLen+C_1
ita_{66}	TR	VM	THiPo-TRLen
ita_{67}	TR	VM	THiPo+C_3
ita_{68}	TR	VM	THiPo
ita_{69}	TR	VM	THiPo-TRLen+C_1+C_4
ita_{70}	TR	VM	THiPo-TRLen-C_5
ita_{71}	BL	VM	BLnID
ita_{72}	BR	VM	BRnID
ita_{73}	BR	VM	f(BLnID,BLLen,BLiBR)

In the exact process of interaction projection, the phenomena replacement could be computed by Algorithm 2. This results in new interactions (from ita_{64} to ita_{73}) as shown in Table VI. Finally, we put the verification problem VP_V into SCADE. The verification costs 833 seconds.

VI. EVALUATION

In this section, we design some experiments to evaluate the two projection methods by answering the following questions:

Q1. What is the performance of these two methods?
Q2. Which scalability issues can be addressed by these two methods?
Q3. How to use these methods more effectively?

We use two sub-problems of ZC, CAL_EOA and CAL_POS for the evaluation. CAL_EOA has been described in the case study section. With inputs start point, direction and distance, CAL_POS outputs a block coordinate. Since a block can only reach the blocks next to it, CAL_POS has to periodically check whether the end point is on current block (it is true only when the distance is reached or there is no more blocks), and if not, check it on next block. This sub-problem has three sub-properties: 1) the input and output coordinates should be on the same track, 2) the distance between input and output coordinates should be the input

distance, and 3) the search direction cannot be changed during the calculation.

In the following experiments, these two sub-problems and their sub-problems are verified repeatedly on machine configurations as shown in Table VII.

TABLE VII
VERIFICATION CONFIGURATION

Sub-problem	CAL_EOA	CAL_POS
CPU	E5 2620 v3	i7 6700
RAM	64GB	16GB
Operating System	WIN7	WIN10

A. Efficiency

Here, the efficiency is measured by the time spent in the verification. An experiment is designed to evaluate the efficiency in two steps. In step 1, we chose CAL_EOA which cannot be verified before projection, to see whether its sub-problem(s) could be verified after the projection. In step 2, we chose CAL_POS which can be verified before, and used it to compare the efficiency of the verification before and after the projection.

In fact, step 1 has been reported in the case study of the previous section. CAL_EOA could not be verified by the built-in verifier in SCADE. It threw an error "Memory Allocation Failure", which was caused by state explosion problem. Even when one uses any one of 8 sub-properties as R', it won't work either. After the projections using the two proposed methods, it could be verified successfully. The results are listed in Table VIII. The PF-based method costs 1103 seconds while the constraints-based method costs 833 seconds. To sum up, it can be seen that the projections have turned the "cannot be verified" sub-problems into "can be verified" ones.

TABLE VIII
EFFICIENCY OF BOTH METHODS

Method	CAL_EOA		CAL_POS	
	time	comparison	time	comparison
no projection	NA	\	33s	\
PF-based	1103s	↓	27s	18%↓
Constraints-based	833s	↓	26s	21%↓

In step 2, firstly we verify CAL_POS using the built-in verifier in SCADE before the projection. It took 33 seconds for all three properties. After PF-based projection, it took 27 seconds for all three properties. The time it took after constraints-based projection was 26 seconds. These numbers can be found in Table VIII.

From Table VIII, one can see that both projection methods can decrease the verification time by about 20%. This result confirmed the theoretical conclusion that these two methods of projection can help decreasing the verification time.

B. Scalability

To generalize the scalability, we studied the relationship between the variable number and the verification time. We designed an experiment using the CAL_POS. Its variable number depends on the number of blocks. The computing formula is $NumV = NumB * 9 + 3$, where $NumV$ is the variable number, and $NumB$ is the block number.

We changed $NumB$ from 1 to 12. For each $NumV$ value, the system were verified three times, and the average time costs are plotted in Figure 5. From the figure, it can be seen that the time cost rises very fast when there are more variables in the environment. Although this was obtained from a sub-problem of ZC, it did imply that more variables require more verification time. In other words, reducing variables number of the system had led to higher efficiency to verify it. The trend in the experiment indicates that the two proposed methods helped verifying all sub-problems in ZC.

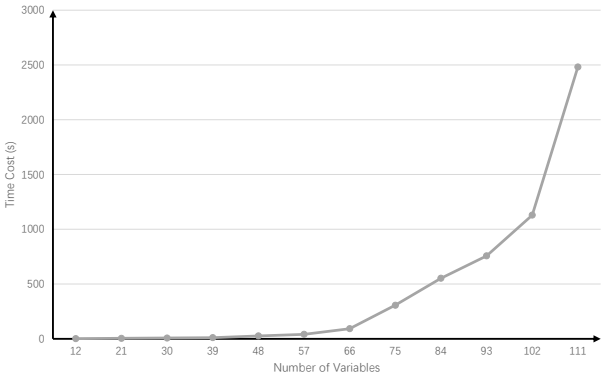


Fig. 5. Relationship between Variables and Verification Time

C. Usage

There are three ways to use the two proposed projection methods: (1) using PF-based method only; (2) using constraint-based method only; and (3) first using PF-based projection then constraints-based projection. We hypothesize that the third way is far more efficient. To validate this hypothesis, we designed a three-step contrast experiment: (1) project the CAL_EOA using the PF-based method; (2) project the CAL_EOA using the constraint-based method; (3) project the same problem first using the PF-based method then the constraint-based method.

After each projection, we obtained (1) 8 subproblems; (2) one problem with a set of 262 constraints; (3) 8 subproblems with 8 sets of constraints. Then we verified them against the 8 sub-properties listed in Table IV, and obtained the time costs.

After the experiment, the results obtained are shown in Figure 6. From this figure, one can see that the constraint-based method achieves a better effect than the PF-based method in all the 8 subproblems. Moreover, the method that uses both projection methods achieved the even better effect compared to the other two alternatives, by reducing at least 21% and 10% time costs respectively.

D. Discussion

Although the proposed methods have achieved expected effects, we found it further challenging to apply them in real

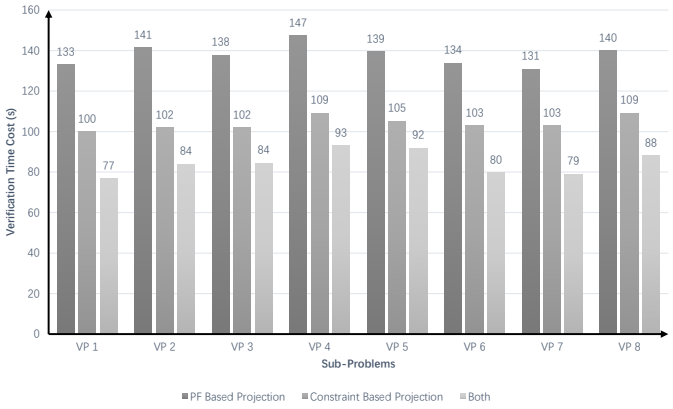


Fig. 6. Verification Time Comparison

industrial settings, i.e., the labour costs and domain knowledge required to organize these constraints and problem diagrams.

The PF-based projection requires problem diagrams of the problem to-be dealt with before-hand. Although we have provided an auxiliary process to obtain problem diagrams [17], novices need extra time to get familiar with the concepts of problem diagrams. In both problem diagrams and transportation systems, even experts also need time to extract corresponding information required. From the perspective of verification, of course, the time cost is worthy the additional effort. For example, as shown in the evaluation, the problem diagram of CAL_EOA took us 1 week after 1 month of learning the ZC specification.

Relatively, the constraints-based method takes constraints and verification problem as inputs. How to get the constraints and how the constraints varies according to different experts are out of the scope of this paper. Even if different experts come up with different constraints, as long as the constraints are correct, our method can use these constraints to optimizes the verification efficiency, achieving good improvements. In the method, generating priority list can still rely on the domain experts. In order to reduce the subjectivity, we have created a priority list generation algorithm for automation. It could also be used for helping or prompting the domain experts if they insist on doing it themselves. In summary, both methods have their applicability conditions. Which one to choose depends on which input condition is more easily satisfied. If both conditions are satisfied, we recommend using that PF-based method first, and the constraints-based method next, which could achieve better performance according to the experiment results found in this case study.

VII. Related Work

Recent research in state spaces simplification of verification problems can be broadly classified into the areas of problem projection, symbolic model checking, statistical model checking, compositional model checking and modeling of .

a) Problem projection: is a kind of decomposition methods in PF approach. It was proposed for decomposing complex problems into overlapping subproblems [10]. However, projections were not automated. To solve this problem, we proposed a scenario-based projection approach [15] which uses scenarios as projection dimensions to treat problems as projection objects, which facilitates automation [18].

Compared with common PF projection approaches, the projection method developed in this paper is aimed at train control verification system and adopts any sub-property as a projection dimension. Customizing to the train transportation domain it becomes possible to effectively reduce the complexity of train control systems.

b) Symbolic model checking: is an optimization verification technique proposed to solve the state explosion problem by McMillan [19].

To reduce the number of states, symbolic model checking does the following. While traversing the verification state spaces, it uses symbolic states instead of concrete states. To organize the traversal spaces, different structures have been proposed. For example, Ordered Binary Decision Diagrams, and conjunctive normal form (CNF) [20].

However, symbolic model checking cannot be applied to our ZC verification problem, which has many numerical computations. Such numerical computations makes it difficult to verify symbolically. In addition, symbolic model checking lacks mature/commercial tools which can be embedded directly into existing design or life cycle tools.

c) Statistical model checking: focuses on stochastic systems and using statistical methods to checking whether the system satisfies a property with a probability higher or lower than a certain threshold [21].

Unlike symbolic model checking, statistical model checking does not traverse the entire state space. In fact, probabilities are estimated in Monte Carlo simulations to control the overhead of verifier in most cases.

Statistical model checking is not suitable for ZC because ZC is a safety critical instead of a stochastic system. Its fault probability allowed by SIL-4 is lower than 10^{-9}, and the safety requirements should be always satisfied.

d) Compositional Model Checking: was initially designed for reducing the complexity of temporal logic model checking in systems composed of many parallel processes [22]. It does not verify a complex program directly, instead it verifies its parallel processes individually and then deduces properties of a composition by checking the properties of individual processes. It has been successfully used in some processor micro-architecture containing most of the features of a modern microprocessor [23][24]. Namjoshi from Bell Laboratories extends it into Parameterized Compositional Model Checking [25] to verify distributed network protocols and shared-memory concurrent programs. Meller also presented a learning-based approach, which generates and verifies behavioral UML systems in the cloud [26].

Our approach is different from the compositional model checking. Instead of focusing on the composition of verification results, we are focusing on the decomposition process. As a result, the verification results of our methods do not need any composition. For any sub-problem which satisfies (or not) a certain sub-property, we can also tell whether the property is satisfied because the sub-problem is actually representative of the problem in our case.

e) Modeling and Safety of Train Control System: Many researches have modeled the train control systems to keep their safety. The most common used approaches are UML-based ones. Normally, they use a subset of UML or UML extension to do modeling. For example, Ossami et al. [27] selected a subset of UML, and investigated a methodology to model guidelines for building certifiable models under railway standards. The model was then transformed to B and FSP [28] in order to validate it with formal semantics. Haxthausen [29] formally developed and verified a distributed railway control system based on RAISE method. They reduced the complexity by separating the system model into a domain model and a controller model.

Other modeling languages are also involved. Hörste et al. [30] modeled a train control system using Petri Nets by the tool Design/CPN. They formally analyzed the capabilities of the system with simulation. Hansen [31] presented a Vienna Development Model (VDM) [32] model of an interlocking system, and described how the model is validated through simulation in ML (programing language). Wang et al [33] provided a three layer model based on stochastic hybrid automata for interlocking systems. Through model simulation with UPPAAL-SMC, they predicted the accidents caused by the equipment faults.

To summarize, above modelings of train control systems, no matter what kinds of modeling language being used, use either simulation or formal verification to ensure their safety. Our approach belongs to the formal verification part. Formal verification has to face the state explosion problem, where our approach can be a pre-process. Simulation does not have this kind of problem. However, it only supports a quick and informal validation of the system model by executing certain execution paths depending on the initial configuration of the model, which is not enough for safety-critical systems [34].

VIII. Conclusions and Future Work

This paper shows that decomposing verification system is an important step to be effective before the formal verification of train control systems. Two projection methods have been proposed for automatically decomposing the ZC verification system. The main contributions are two folds:

- The Problem Frames (PF)-based projection is defined using sub-properties as projection dimensions. The ZC verification system is modeled by a problem diagram by following the PF approach. Sub-properties relate every problem element together thereby forming a subproblem.
- The constraints-based projection is defined with a set of variables as projection dimensions. The variables are chosen from a set of constraints. An algorithm has been developed for automatically generating these variables.

We have applied these two methods into real industrial cases and done some experiments. By comparing the time costs of the verification before and after using these methods, the results have shown a great decrease in verification time costs. Especially in some cases, despite the state explosion problem arises before the projection but its sub-problems can be still verified. This makes a great advance in ZC verification work.

For future work we are considering applying the methods to other systems of CBTC. Since constraints-based projection seems to be more generalizable found by this case study, we aim to use this method and to elicit the relationships between sub-properties and sub-systems used in the PF-based projection from these systems, as well as from the relevant industrial standards.

Acknowledgment

The authors would like to thank Liangyu Chen and Min Zhang from East China Normal University for their great help during the development of the algorithms implementation and proof of the hypotheses.

References

[1] P. Behm, P. Benoit, A. Faivre, and J.-M. Meynadier, "Meteor: A successful application of b in a large project," in *International Symposium on Formal Methods*. Springer, 1999, pp. 369–387.

[2] E. CENELEC, "50128," *Railway applications-Communication, Signaling and Processing Systems-Software for Railway Control and Protection Systems*, 2011.

[3] N. C. EN, "50129: Railway application–communications, signaling and processing systems–safety related electronic systems for signaling," *British Standards*, 2003.

[4] A. Cimatti, F. Giunchiglia, G. Mongardi, D. Romano, F. Torielli, and P. Traverso, "Formal verification of a railway interlocking system using model checking," *Formal Aspects of Computing*, vol. 10, no. 4, pp. 361–380, 1998.

[5] P. A. Abdulla, J. Deneux, G. Stålmarck, H. Ågren, and O. Åkerlund, "Designing safe, reliable systems using scade," in *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. Springer, 2004, pp. 115–129.

[6] T. Le Sergent, "Scade: A comprehensive framework for critical system and software engineering," in *International SDL Forum*. Springer, 2011, pp. 2–3.

[7] S. Yang, "Modeling and safety verification of zone controller in cbtc with uml [ms. thesis]," *Beijing: Beijing Jiaotong University*, 2008.

[8] "Ieee standard for communications-based train control (cbtc) performance and functional requirements," *IEEE Std 1474.1-2004*, pp. 01–45, 2004.

[9] J. Xu, X. Chen, T. Zhou, Z. Yuan, and K. Huang, "Decomposing automatic train control verification system with projection," in *Software Engineering Conference (APSEC), 2015 Asia-Pacific*. IEEE, 2015, pp. 301–308.

[10] M.Jackson, *Problem Frames: Analyzing and Structuring software development problems*. Addison-Wesley, 2001.

[11] X. Chen and Z. Jin, "Capturing software requirements from the expected interactions between the software and its environment: an ontology based approach," *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 1, pp. 15–39, 2016.

[12] "Ieee recommended practice for communications-based train control (cbtc) system design and functional allocations," *IEEE Std 1474.3-2008*, pp. c1–117, 2008.

[13] Esterel-Technologies, *Scade Language Reference Manual*, 2013.

[14] K. L. McMillan, *Symbolic Model Checking: An Approach to the State Explosion Problem*, 1992.

[15] Z. Jin, X. Chen, and D. Zowghi, "Performing projection in problem frames using scenarios," in *Software Engineering Conference, 2009. APSEC'09. Asia-Pacific*. IEEE, 2009, pp. 249–256.

[16] L. Han, J. Liu, T. Zhou, J. Sun, and X. Chen, "Safety requirements specification and verification for railway interlocking systems," in *Computer Software and Applications Conference (COMPSAC), 2016 IEEE 40th Annual*, vol. 1. IEEE, 2016, pp. 335–340.

[17] X. Chen, Z. Jin, and L. Yi, "An ontology of problem frames for guiding problem frame specification," in *International Conference on Knowledge Science, Engineering and Management*. Springer, 2007, pp. 384–395.

[18] X. Chen, B. Yin, and Z. Jin, "Dptool: A tool for guiding the problem description and the problem projection," in *the 18th IEEE International RequirementsEngineering Conference*, 2010, pp. 401–402.

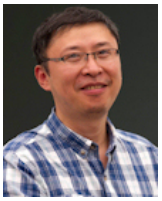
[19] K. L. McMillan, "Symbolic model checking," in *Symbolic Model Checking*. Springer, 1993, pp. 25–60.

[20] ——, "Applying sat methods in unbounded symbolic model checking," in *CAV*, vol. 2. Springer, 2002, pp. 250–264.
[21] H. L. Younes, "Verification and planning for stochastic processes with asynchronous events," DTIC Document, Tech. Rep., 2005.
[22] E. M. Clarke, D. E. Long, and K. L. McMillan, "Compositional model checking," in *Logic in Computer Science, 1989. LICS'89, Proceedings., Fourth Annual Symposium on*. IEEE, 1989, pp. 353–362.
[23] R. Jhala and K. L. McMillan, "Microarchitecture verification by compositional model checking," in *International Conference on Computer Aided Verification*. Springer, 2001, pp. 396–410.
[24] K. McMillan, "Parameterized verification of the flash cache coherence protocol by compositional model checking," *Correct hardware design and verification methods*, pp. 179–195, 2001.
[25] K. S. Namjoshi and R. J. Trefler, "Parameterized compositional model checking," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2016, pp. 589–606.
[26] Y. Meller, O. Grumberg, and K. Yorav, "Learning-based compositional model checking of behavioral uml systems," in *International Workshop on Formal Aspects of Component Software*. Springer, 2015, pp. 275–293.
[27] D. D. O. Ossami, J.-M. Mota, L. Thiry, J.-M. Perronne, J.-L. Boulanger, and G. Mariano, "A method to model guidelines for developing railway safety-critical systems with uml." in *ICSOFT (SE)*, 2007, pp. 236–243.
[28] J. Magee and J. Kramer, *State models and java programs*.
[29] A. E. Haxthausen and J. Peleska, "Formal development and verification of a distributed railway control system," *IEEE Transactions on Software Engineering*, no. 8, pp. 687–701, 2000.
[30] M. M. zu Hörste and E. Schnieder, "Modelling and simulation of train control systems using petri nets," in *FMrail workshop*, vol. 3, 1999.
[31] K. M. Hansen, "Validation of a railway interlocking model," in *International Symposium of Formal Methods Europe*. Springer, 1994, pp. 582–601.
[32] C. B. Jones, *Systematic software development using VDM*. Citeseer, 1990, vol. 2.
[33] Y. Wang, W. Zhong, X. Chen, and J. Liu, "Predicting accidents in interlocking systems: An sha model-based approach," *International Journal of Performability Engineering*, vol. 13, no. 6, pp. 897–912, 2017.
[34] D. Wu and E. Schnieder, "Scenario-based system design with colored petri nets: an application to train control systems," *Software & Systems Modeling*, vol. 17, no. 1, pp. 295–317, 2018.

Zhengheng Yuan is a PhD. Student at East China Normal University. He received his BS in Mathematics and Applied Mathematics from East China University of Science and Technology in 2012. His research interests include formal methods and hybrid systems.

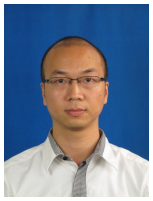
Xiaohong Chen is currently an associate professor at East China Normal University. She received PhD degree from the Academy of Mathematics and Systems Science, Chinese Academy of Sciences in 2010. Her research interests include requirements engineering, model driven development, and cyber-physical systems.

Jing Liu is currently a professor of computer science at East China Normal University (ECNU), China. She is also the vice director of Computing Theories Institute, ECNU. In recent years, she is working on the area of model driven architecture. Now her work focuses on the design of real-time embedded systems and cyber physical system.



Yijun Yu is a Senior Lecturer in Computing at The Open University, UK since 2006. He is interested in developing automated, efficient and scalable software techniques and tools to better support human activities in software engineering. His research on requirements-driven adaptation receives a 10 Year Most Influential Paper (CASCON'06), 3 Best Paper (iRENIC'16, IEEE TrustCom'14, ACM EICS'13), 3 Distinguished Paper (IEEE RE'11, BCS'08, ACM SigSoft ASE'07), and a Best Tool Demo Paper (RE'13) awards.

Haiying Sun was born in 1976. She received the master's degree in computer science and technology from the NanJing University Science and Technology in 2001. She earned her PhD. degree in Formal Method From East China Normal University in 2017. She is now a lecturer at East China Normal University. Her research interests include formal method, automatic test generation, system simulation and model-driven engineering.



Tingliang Zhou is currently a director of R&D Institute of CASCO Signal Ltd. In recent years, he is working on the area of safety critical software verification and validation. Now his work focuses on the design of automatic train control system.

Zhi Jin is a full professor of software engineering at Peking University. She's also the vice director of the Key Laboratory of High Confidence Software Technologies (Peking University) of the Ministry of Education of China. Her research interests include requirements engineering and knowledge engineering. Jin received a PhD in computer science from the Changsha Institute of Technology. Contact her at zhijin@pku.edu.cn.